Lab1 – Android Development Environment

Setting up the ADT, Creating, Running and Debugging Your First Application

Objectives:

Familiarize yourself with the Android Development Environment

Important Note: This class has many students with a wide range of previous experience. Some students are fairly new to object-oriented programming (OOP). Some have OOP experience, but are new to Android. Still others have some Android experience already, and want to just freshen up their knowledge.

Because of this, I'm not expecting that everyone can finish this entire lab. I suggest that you set a time limit for yourself, say 1 hour. Work through what you can in that time and then stop and take a break. If you later feel that you have some more time for this Lab, then repeat the process. Again – don't feel that you need to finish everything in this lab. That's not the goal here.

Specifically, if you are fairly new to programming, you should try to complete Parts 1 – 4 below. If you are familiar with programming and programming environments, you should try to complete parts 1 – 6 below.

This lab contains the following Parts.

1. Set up Android Studio.
2. Create a new Android application.
3. Create an Android Virtual Device and start the Android Emulator.
4. Run the application you created in Part 2.
5. Import an application project.
6. Debug an Android application.

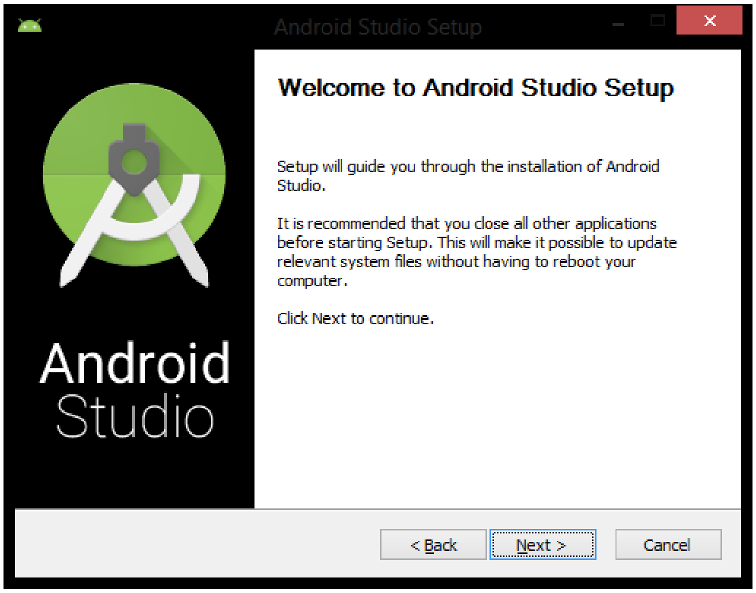
Additional helpful information can be found on the Android Developer website:

- https://developer.android.com/studio/index.html
- https://developer.android.com/training/basics/firstapp/creating-project.html
- https://developer.android.com/studio/run/managing-avds.html
- https://developer.android.com/training/basics/firstapp/running-app.html

Part 1 – Setting Up Android Studio.

In this part you will download and install Android Studio which will be the Integrated Development Environment (IDE) used for this course. For the purposes of this document, we installed Android Studio version 3.1.4 (the current latest stable release as of 7/23/2018) on a Mac running High Sierra. All screenshots correspond to that environment.

1. Download Android Studio from https://developer.android.com/studio/index.html . Click on 'Download Android Studio'.
2. Open the executable file android-studio-<xxx>.
3. Once the setup loads, you will see the Welcome Screen.



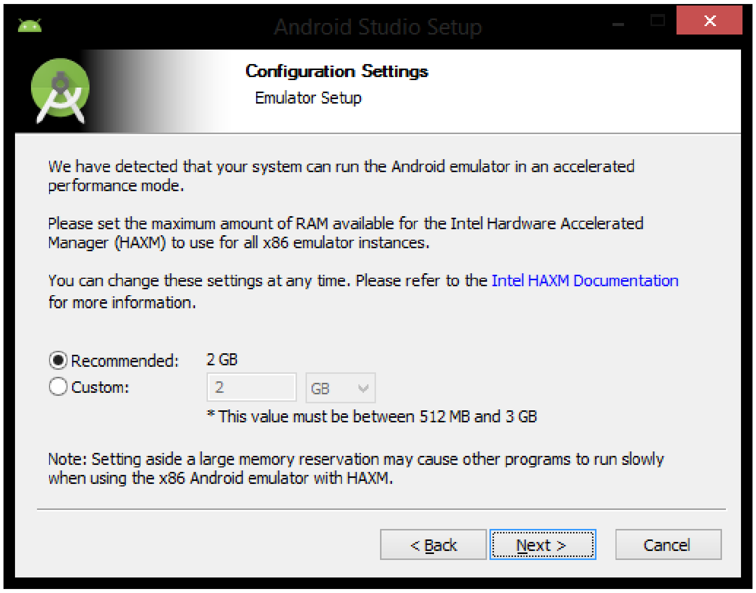
Welcome to Android Studio Setup

Setup will guide you through the installation of Android Studio.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

4. Click 'Next >' on the Welcome Screen.
5. When choosing components, ensure all of the checkboxes are checked in for each component to install. Once you are done, click 'Next >'.
6. Agree to the Android Studio and the Intel HAXM License Agreements after reading them.
7. Verify the install locations meet the installation requirements and click 'Next >'.
8. You may or may not see the emulator setup settings, just click 'Next >' after selecting the RAM size.



Configuration Settings
Emulator Setup

We have detected that your system can run the Android emulator in an accelerated performance mode.

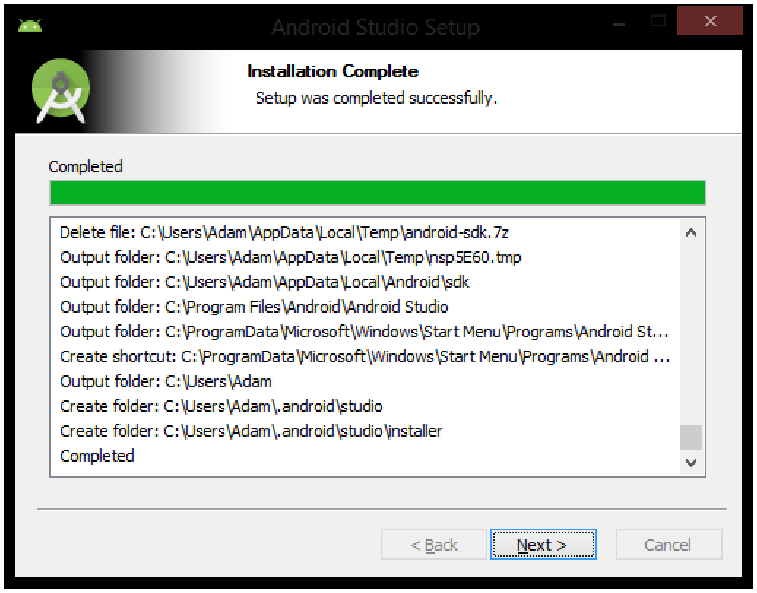
Please set the maximum amount of RAM available for the Intel Hardware Accelerated Manager (HAXM) to use for all x86 emulator instances.

You can change these settings at any time. Please refer to the Intel HAXM Documentation for more information.

● Recommended: 2 GB
○ Custom: 2 GB
* This value must be between 512 MB and 3 GB

Note: Setting aside a large memory reservation may cause other programs to run slowly when using the x86 Android emulator with HAXM.

9. Finally, click ‘Install’. You will see which operations are currently running in the installation process and a progress bar displaying their progress.
10. Once the installation process is finished click ‘Next >’.

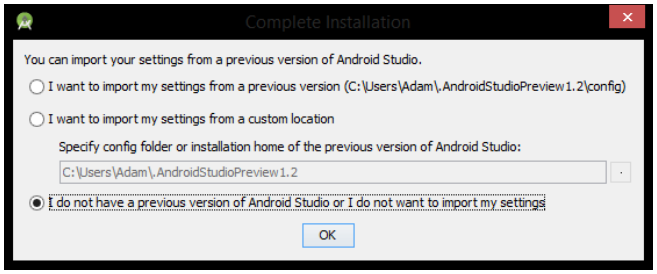


Installation Complete
Setup was completed successfully.

Completed

```
Delete file: C:\Users\Adam\AppData\Local\Temp\android-sdk.7z
Output folder: C:\Users\Adam\AppData\Local\Temp\nsp5E60.tmp
Output folder: C:\Users\Adam\AppData\Local\Android\sdk
Output folder: C:\Program Files\Android\Android Studio
Output folder: C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Android St...
Create shortcut: C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Android ...
Output folder: C:\Users\Adam
Create folder: C:\Users\Adam\.android\studio
Create folder: C:\Users\Adam\.android\studio\installer
Completed
```

11. Android Studio is now set up. Check on ‘Start Android Studio’ and click ‘Finish’.
12. You will see the Complete Installation screen below.
13. If you had a previous version of Android Studio installed prior, then check either the first or second radio box. Otherwise, check the last radio box and hit ‘OK’.



You can import your settings from a previous version of Android Studio.

○ I want to import my settings from a previous version (C:\Users\Adam\.AndroidStudioPreview1.2\config)

○ I want to import my settings from a custom location

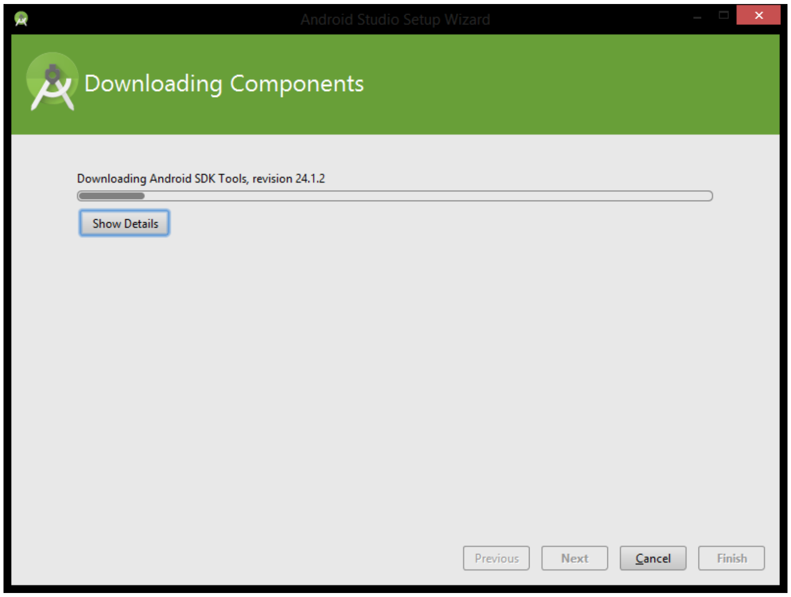
Specify config folder or installation home of the previous version of Android Studio:

C:\Users\Adam\.AndroidStudioPreview1.2

● I do not have a previous version of Android Studio or I do not want to import my settings

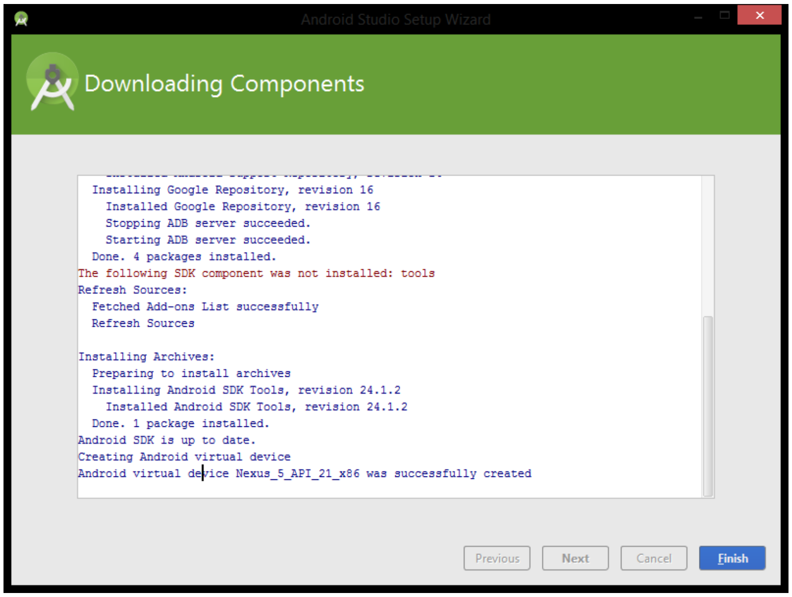
14. As Android Studio starts, the splash screen will appear.

15. After the splash screen you may see some additional setup operations run, such as downloading components.



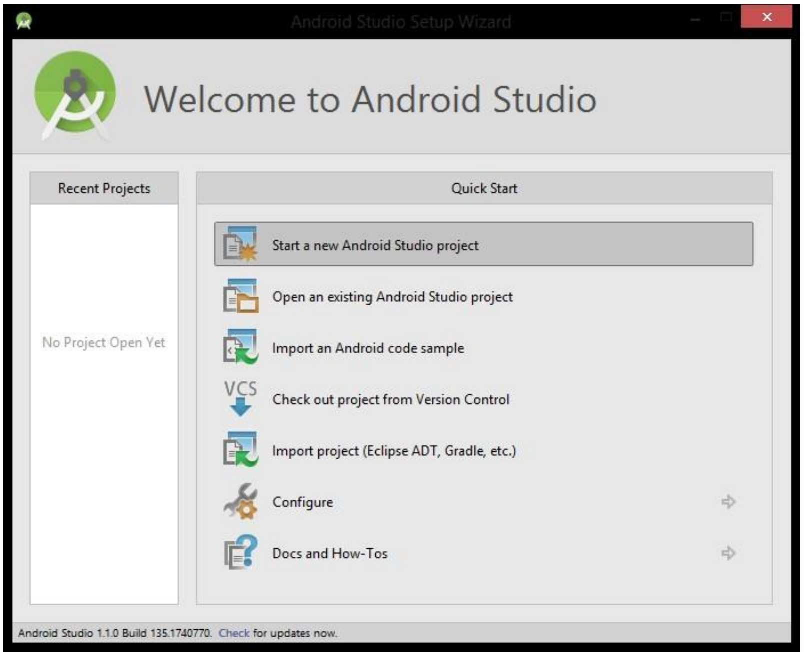
Downloading Android SDK Tools, revision 24.1.2

16. Once it is finished, click 'Finish'.



```
  Installing Google Repository, revision 16
    Installed Google Repository, revision 16
    Stopping ADB server succeeded.
    Starting ADB server succeeded.
  Done. 4 packages installed.
The following SDK component was not installed: tools
Refresh Sources:
  Fetched Add-ons List successfully
  Refresh Sources

Installing Archives:
  Preparing to install archives
  Installing Android SDK Tools, revision 24.1.2
    Installed Android SDK Tools, revision 24.1.2
  Done. 1 package installed.
Android SDK is up to date.
Creating Android virtual device
Android virtual device Nexus_5_API_21_x86 was successfully created
```

17. Welcome to Android Studio! In the next part we will start our first project.

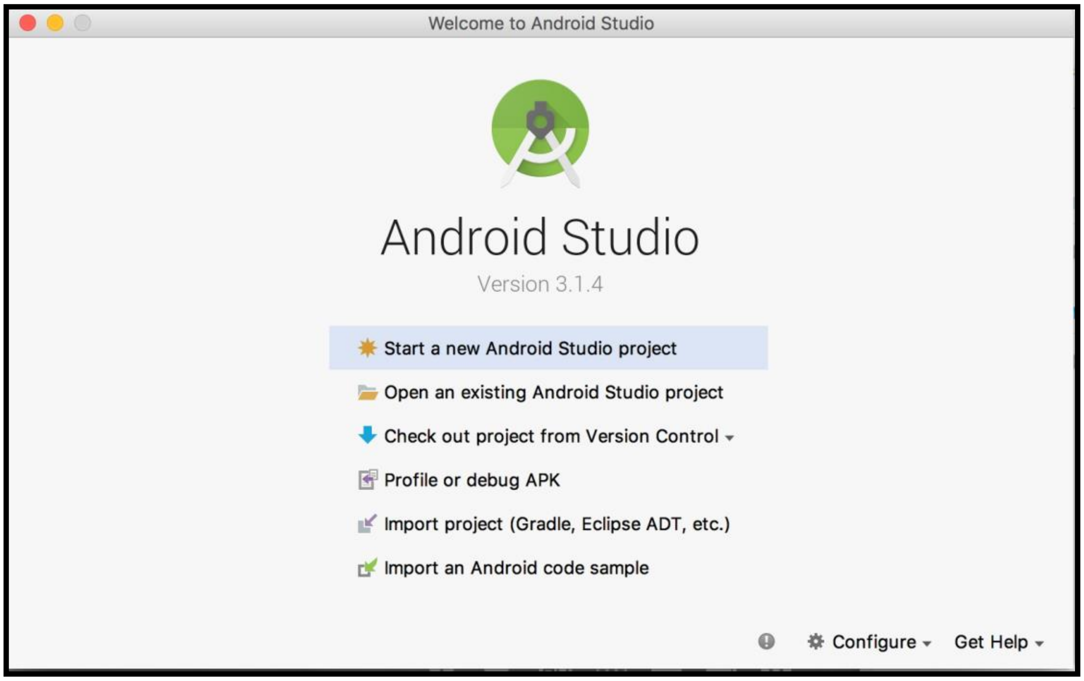

Android Studio Setup Wizard
Welcome to Android Studio
Recent Projects
No Project Open Yet
Quick Start
Start a new Android Studio project
Open an existing Android Studio project
Import an Android code sample
VCS
Check out project from Version Control
Import project (Eclipse ADT, Gradle, etc.)
Configure
Docs and How-Tos

Android Studio 1.1.0 Build 135.1740770. Check for updates now.

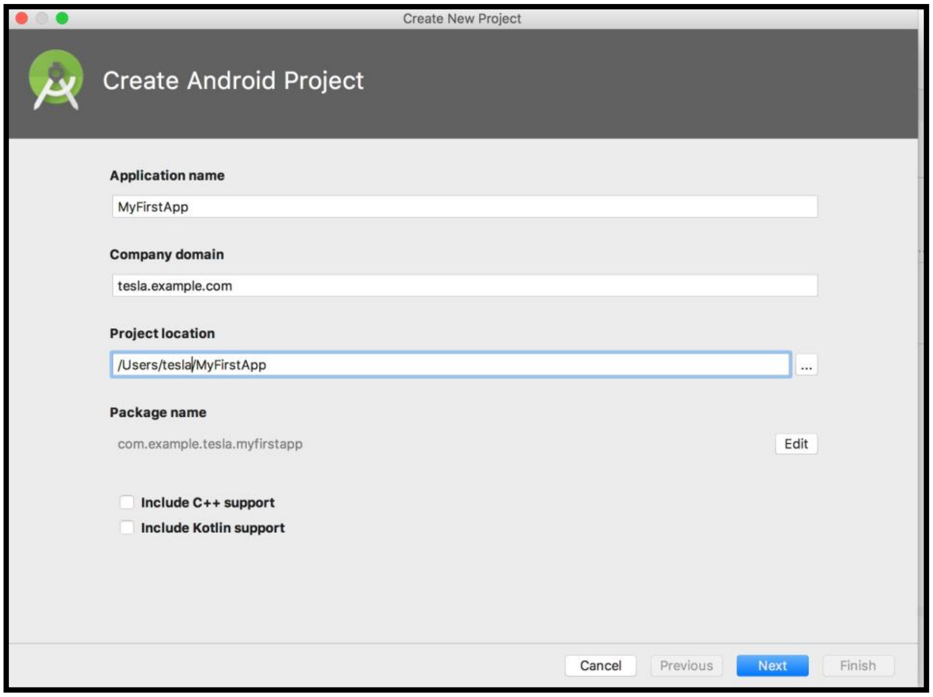
Part 2 – Creating A New Project

In this part you will create a simple Android application that displays the words, "Hello World!"

1. At the Welcome Screen, click on ‘Start a new Android Studio project’.


Welcome to Android Studio
Android Studio
Version 3.1.4
Start a new Android Studio project
Open an existing Android Studio project
Check out project from Version Control
Profile or debug APK
Import project (Gradle, Eclipse ADT, etc.)
Import an Android code sample
Configure
Get Help

2. Enter the application name ‘MyFirstApp’ and note where the project is located. I am creating my project inside of my home directory.



Application name: MyFirstApp

Company domain: tesla.example.com

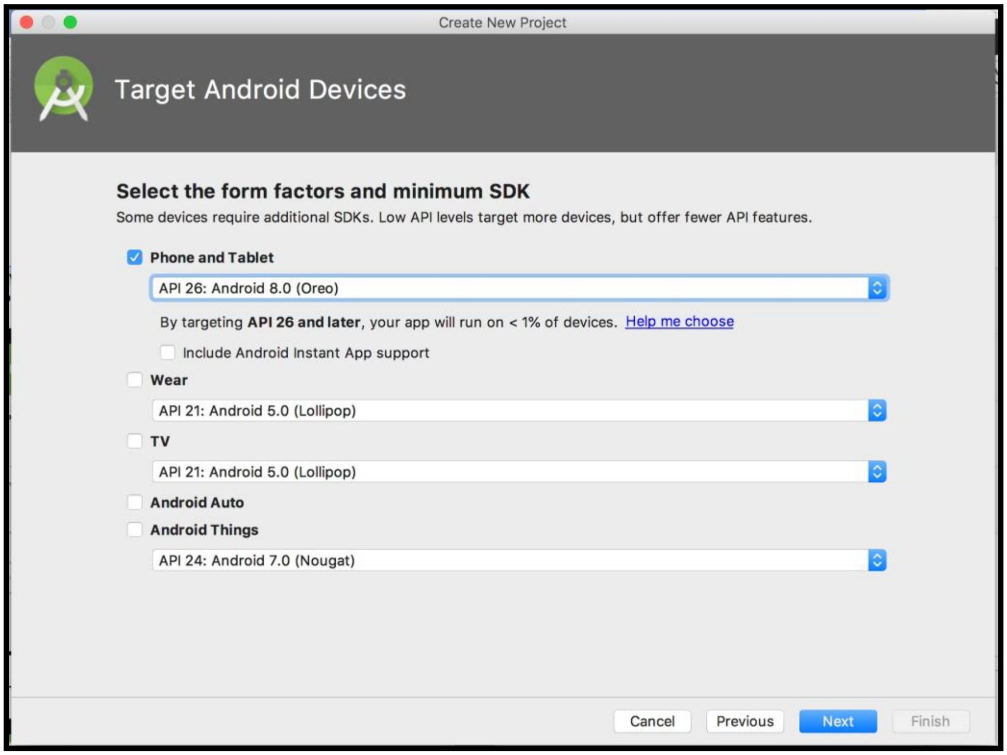
Project location: /Users/tesla/MyFirstApp

Package name: com.example.tesla.myfirstapp

□ Include C++ support

□ Include Kotlin support

3. Select which devices you would like your app to run on. For now we will be working with 'Phone and Tablet'. Make sure to set the Minimum SDK version to API 26 for this course.



Create New Project

Target Android Devices

Select the form factors and minimum SDK

Some devices require additional SDKs. Low API levels target more devices, but offer fewer API features.

☑ Phone and Tablet

API 26: Android 8.0 (Oreo)

By targeting **API 26 and later**, your app will run on < 1% of devices. Help me choose

□ Include Android Instant App support

□ Wear

API 21: Android 5.0 (Lollipop)

□ TV

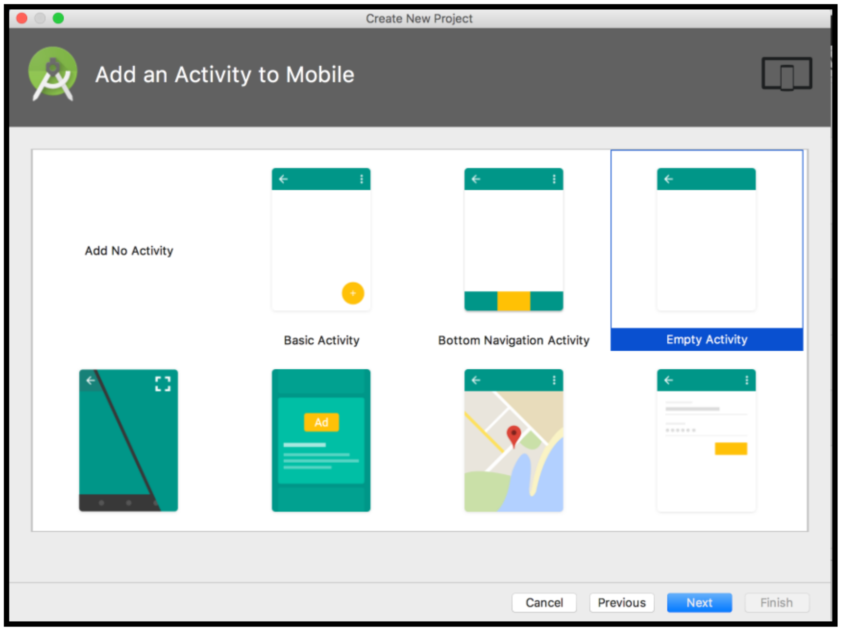
API 21: Android 5.0 (Lollipop)

□ Android Auto

□ Android Things

API 24: Android 7.0 (Nougat)

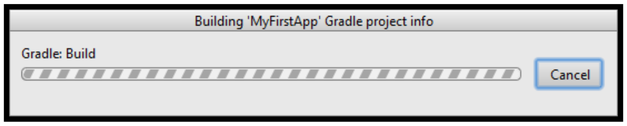
4. Select 'Empty Activity' and click 'Next'.



Create New Project

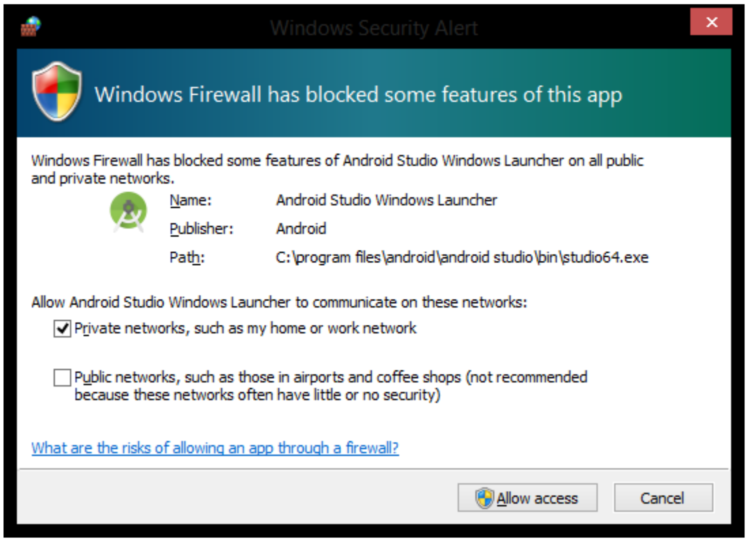
Add an Activity to Mobile

5. In the next window, leave all the settings as default, and then click Finish.
6. Android Studio will now create the project and build it.



Gradle: Build

7. You may see a security alert if you are on Windows, click ‘Allow access’ to continue.



Windows Firewall has blocked some features of this app

Windows Firewall has blocked some features of Android Studio Windows Launcher on all public and private networks.

Name: Android Studio Windows Launcher

Publisher: Android

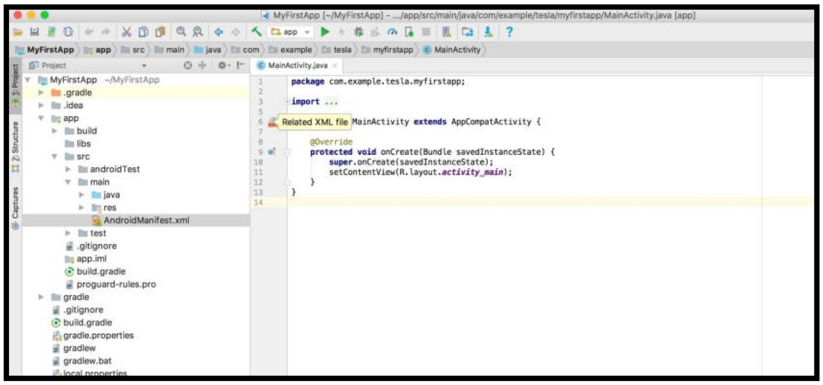
Path: C:\program files\android\android studio\bin\studio64.exe

Allow Android Studio Windows Launcher to communicate on these networks:

☑ Private networks, such as my home or work network

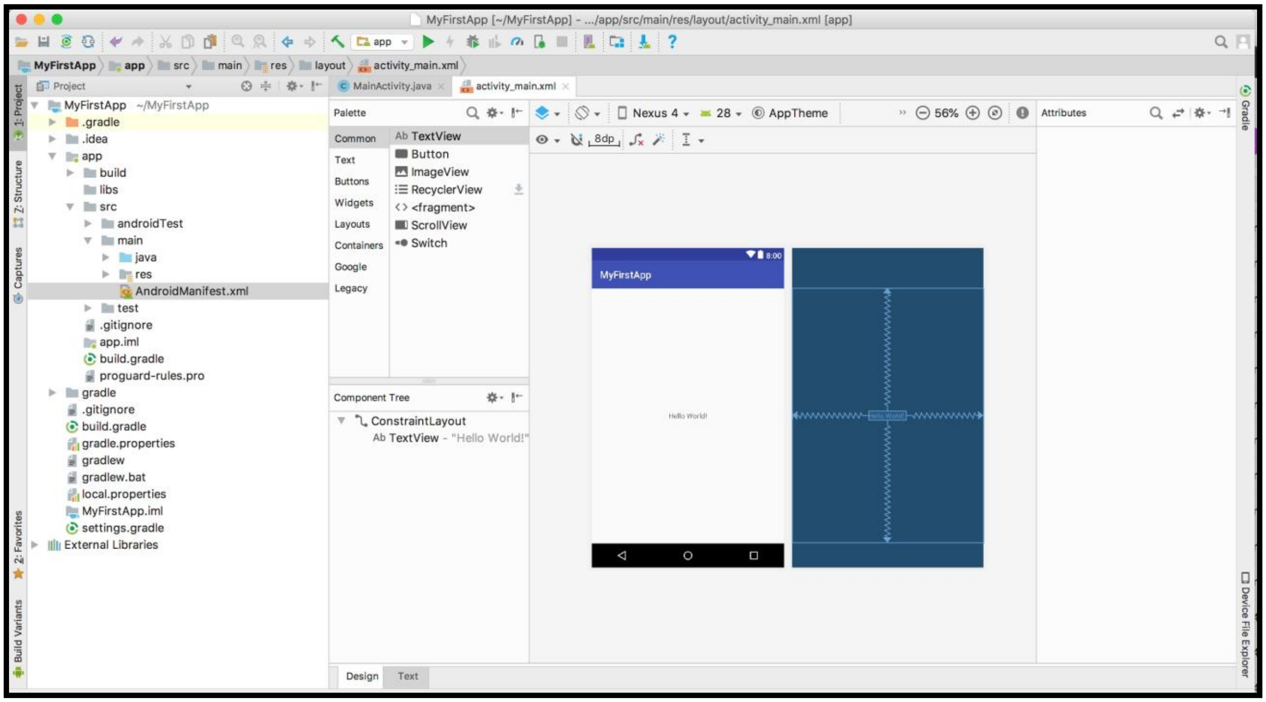
☐ Public networks, such as those in airports and coffee shops (not recommended because these networks often have little or no security)

8. Once the Android Studio IDE fully loads, click on the ‘Related XML file’ icon you will see by hovering over the icon in the middle of the window.



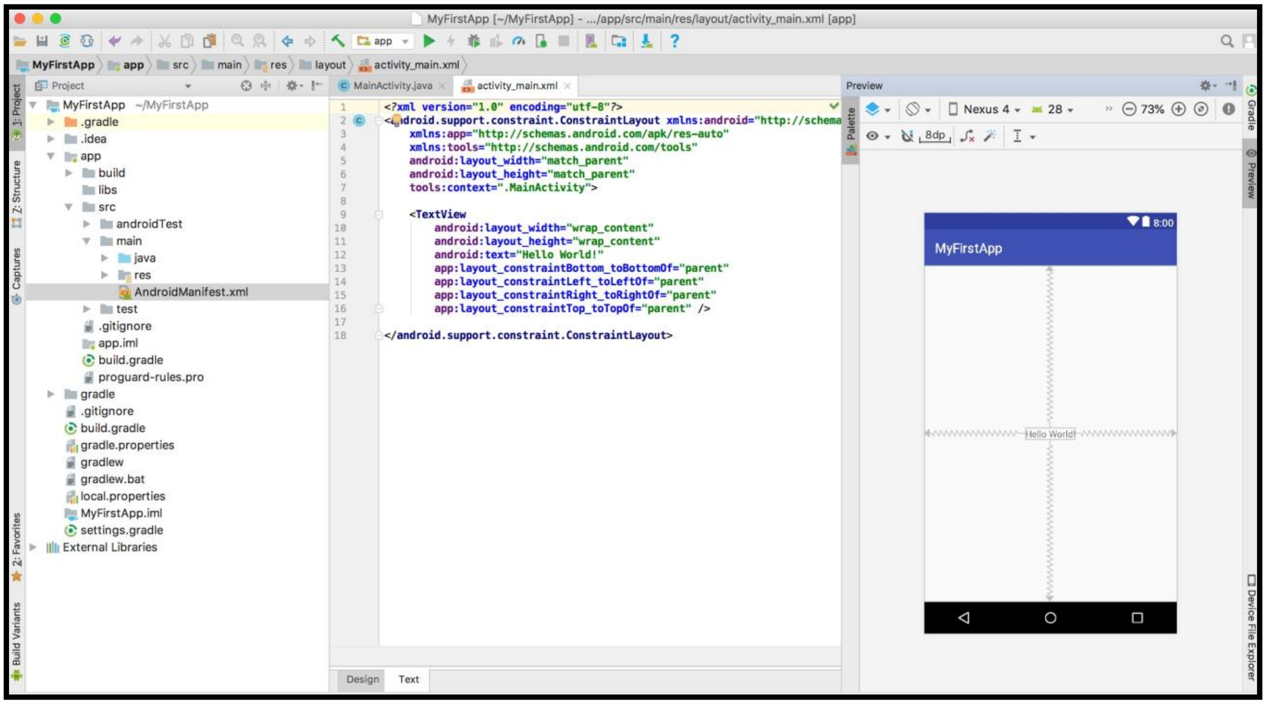
```
package com.example.tesla.myfirstapp;

import ...

                    MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

This will now open the following view inside of Android Studio:



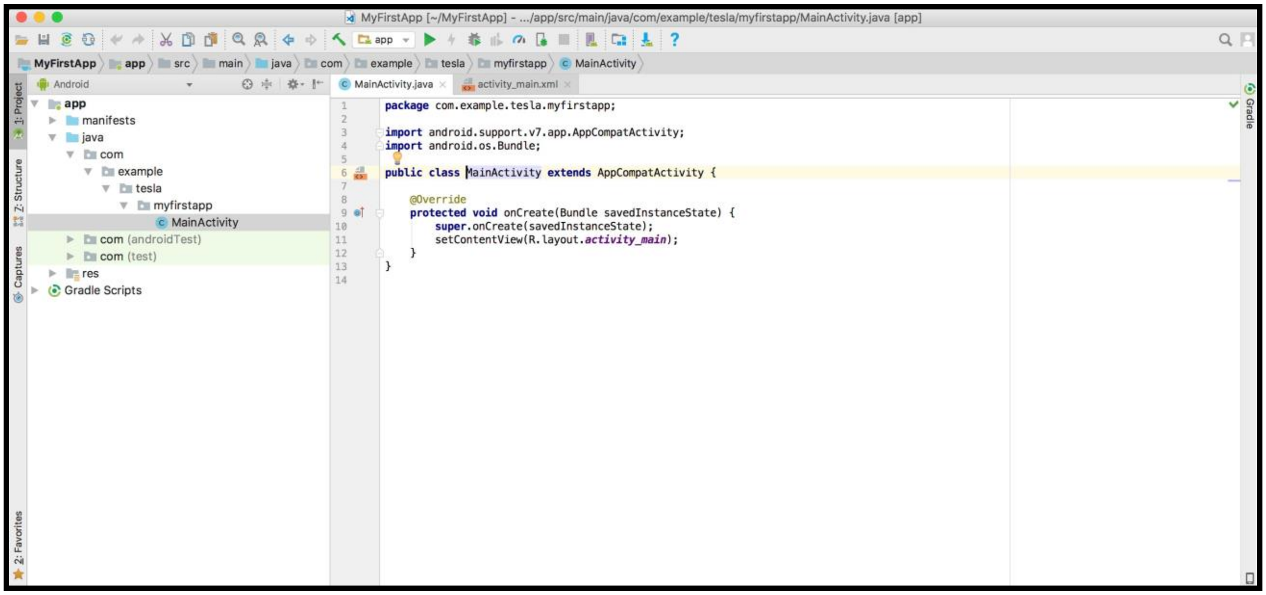
9. The screen you now see is the Design View of the activity_main.xml file. You can already see the words “Hello World!” on the App’s User Interface.

10. If you click on the Text tab on the bottom you can see the layout file underlying the user interface.



```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schema
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

11. To view the backing code for this activity, double click on ‘MainActivity’ inside of the Project directory tree. This file is located in: ‘java’ > ‘com.example.<user_name>.myfirstapp’ > ‘MainActivity’.



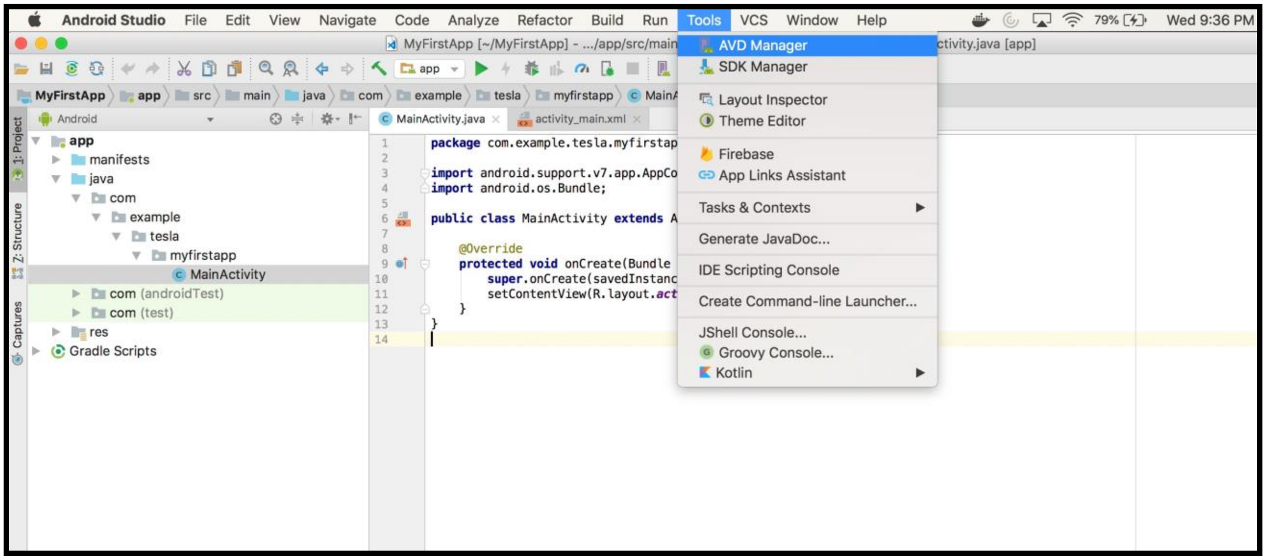
```
package com.example.tesla.myfirstapp;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

In Part 4 we will show you how to run this app in the Android Emulator.

Part 3 – Using the Emulator

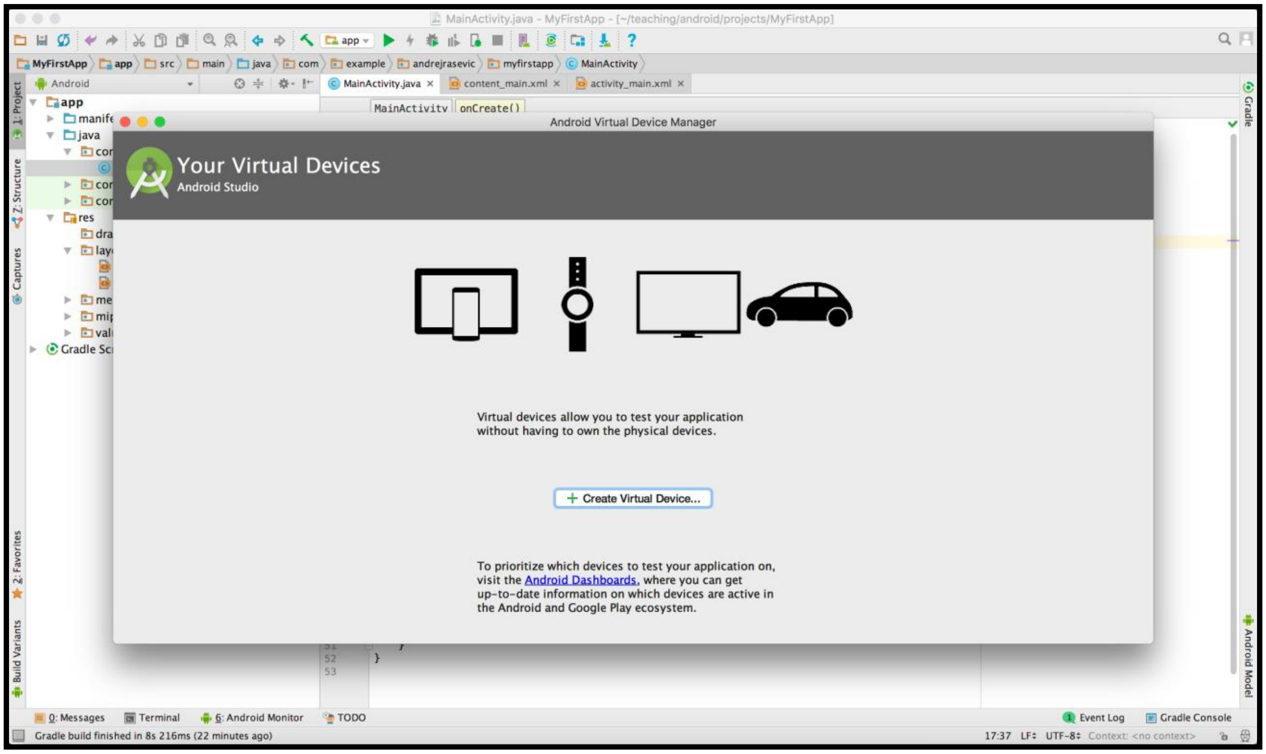
In this part you will learn how to set up and use the Android Emulator.

1. First start up the Android Virtual Device Manager. You can do that by selecting Tools > AVD Manager from the Android Studio menu bar.



```
package com.example.tesla.myfirstap

import android.support.v7.app.AppCo
import android.os.Bundle;

public class MainActivity extends A

    @Override
    protected void onCreate(Bundle
        super.onCreate(savedInstanc
        setContentView(R.layout.act
    }
}
```

2. A new dialog box will pop up.

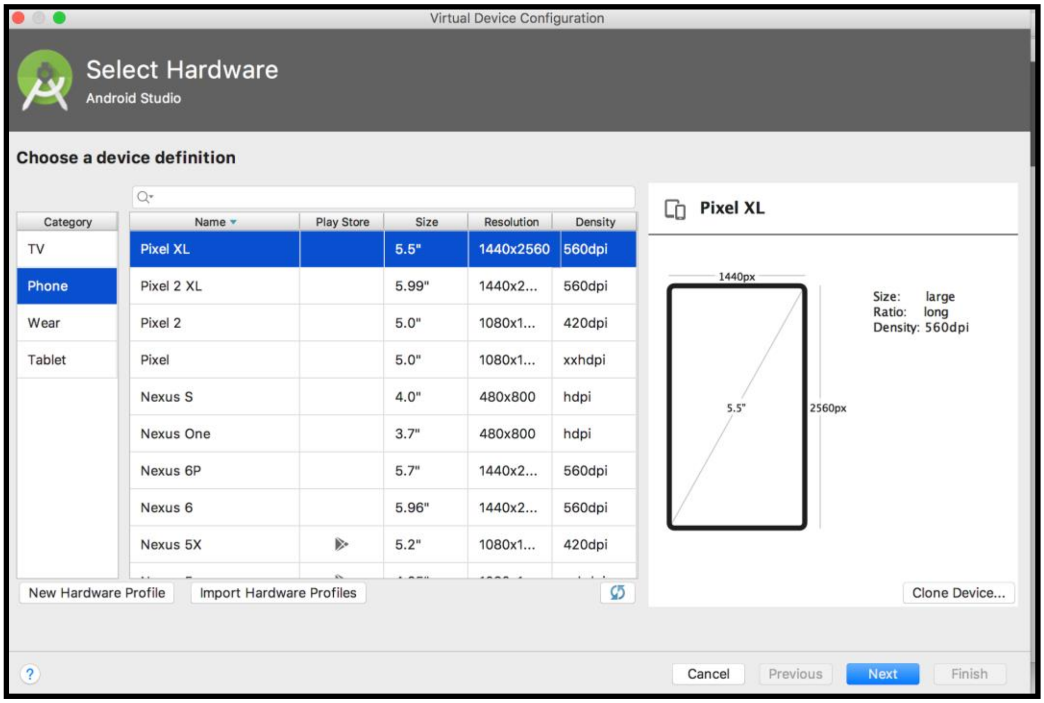

MainActivity.java - MyFirstApp - [~/teaching/android/projects/MyFirstApp]
MyFirstApp app src main java com example andrejrasevic myfirstapp MainActivity
Android
MainActivity.java
content_main.xml
activity_main.xml
Android Virtual Device Manager
Your Virtual Devices
Android Studio
+ Create Virtual Device...
0: Messages
Terminal
6: Android Monitor
TODO
Event Log
Gradle Console

Virtual devices allow you to test your application without having to own the physical devices.

To prioritize which devices to test your application on, visit the Android Dashboards, where you can get up-to-date information on which devices are active in the Android and Google Play ecosystem.

Gradle build finished in 8s 216ms (22 minutes ago)

3. Click "Create Virtual Device" to create a new Android Virtual Device (AVD).



Choose a device definition

Category
TV
Phone
Wear
Tablet

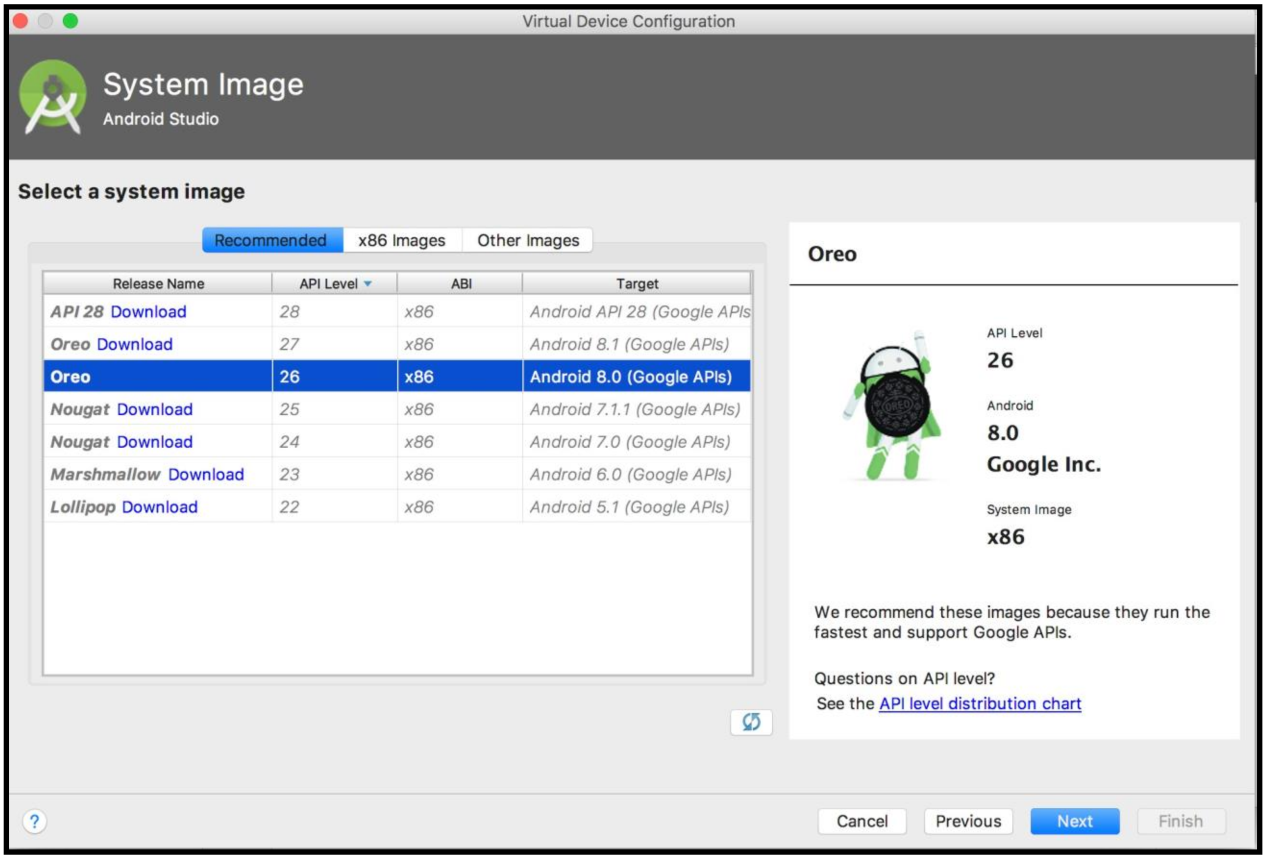
Name	Play Store	Size	Resolution	Density
Pixel XL		5.5"	1440x2560	560dpi
Pixel 2 XL		5.99"	1440x2...	560dpi
Pixel 2		5.0"	1080x1...	420dpi
Pixel		5.0"	1080x1...	xxhdpi
Nexus S		4.0"	480x800	hdpi
Nexus One		3.7"	480x800	hdpi
Nexus 6P		5.7"	1440x2...	560dpi
Nexus 6		5.96"	1440x2...	560dpi
Nexus 5X	▶	5.2"	1080x1...	420dpi

Pixel XL

Size: large
Ratio: long
Density: 560dpi

4. Another dialog box will pop up displaying various pre-made AVD templates. To simulate the same environment we will be testing your code in, select phone in the left panel and then Pixel XL in the main panel. Now click ‘Next’.

5. Select the appropriate System Image for the virtual machine. All of the class projects will be tested against API level 26. If you haven’t downloaded that already, make sure to download it now, by clicking on the “Download” link. As you can see in the screenshot, I have already downloaded API 26 in my environment. Once it has downloaded click ‘Next’.



System Image

Android Studio

Select a system image

Release Name	API Level ▾	ABI	Target
API 28 Download	*28*	*x86*	*Android API 28 (Google APIs*
Oreo Download	*27*	*x86*	*Android 8.1 (Google APIs)*
Oreo	26	x86	Android 8.0 (Google APIs)
Nougat Download	*25*	*x86*	*Android 7.1.1 (Google APIs)*
Nougat Download	*24*	*x86*	*Android 7.0 (Google APIs)*
Marshmallow Download	*23*	*x86*	*Android 6.0 (Google APIs)*
Lollipop Download	*22*	*x86*	*Android 5.1 (Google APIs)*

Oreo

API Level
26

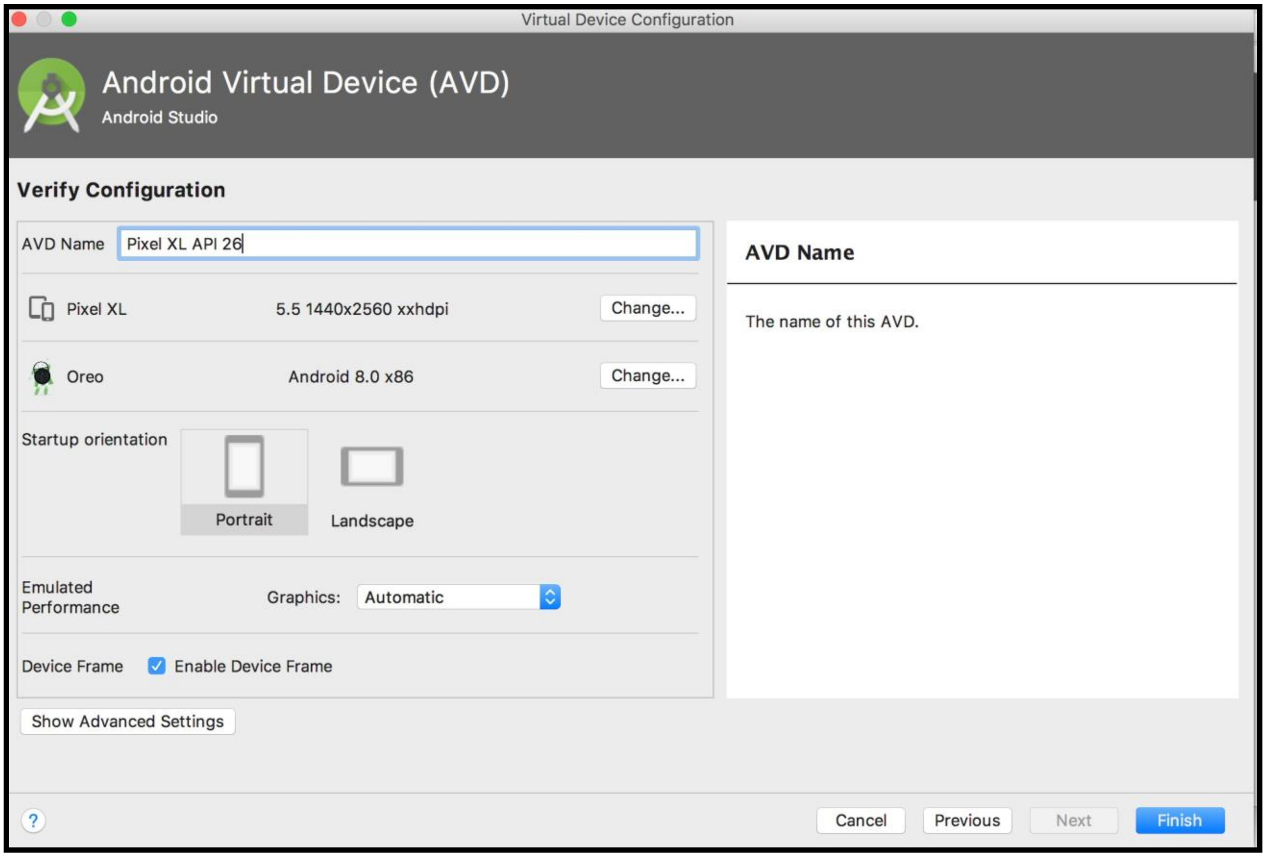
Android
8.0
Google Inc.

System Image
x86

We recommend these images because they run the fastest and support Google APIs.

Questions on API level?
See the API level distribution chart

Select Portrait and the click ‘Finish’.



Virtual Device Configuration

Android Virtual Device (AVD)
Android Studio

Verify Configuration

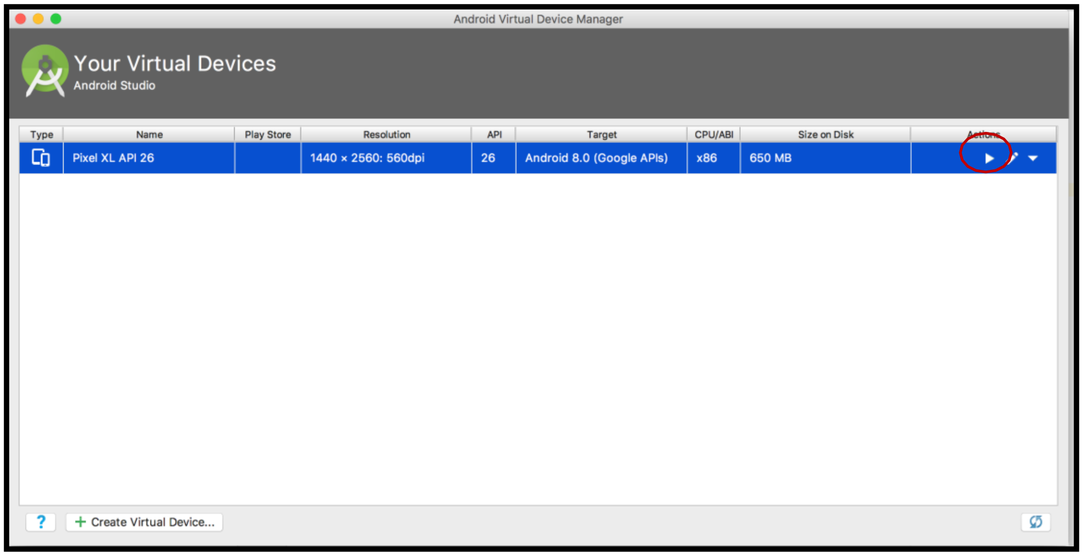
AVD Name	Pixel XL API 26	
Pixel XL	5.5 1440x2560 xxhdpi	Change...
Oreo	Android 8.0 x86	Change...
Startup orientation	Portrait	Landscape
Emulated Performance	Graphics: Automatic	
Device Frame	☑ Enable Device Frame	

Show Advanced Settings

AVD Name

The name of this AVD.

6. Now click on the ‘Play’ icon to start the emulator, after clicking on it you can close the Android Virtual Device Manager.



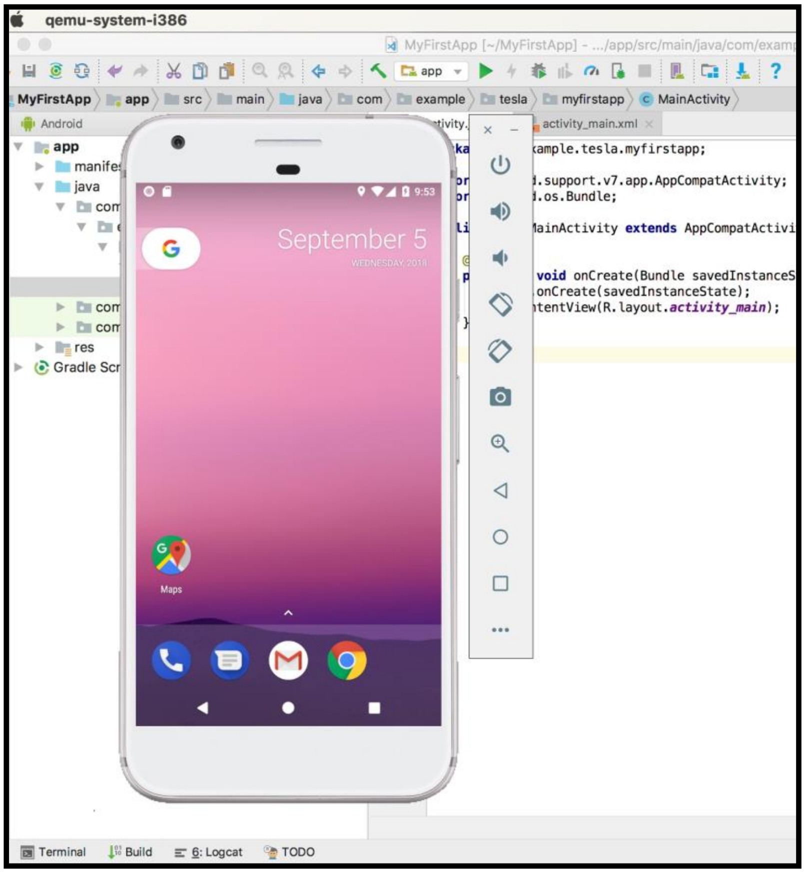
Android Virtual Device Manager

Your Virtual Devices
Android Studio

Type	Name	Play Store	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Pixel XL API 26		1440 × 2560: 560dpi	26	Android 8.0 (Google APIs)	x86	650 MB	▶ ✎ ▾

7. As the emulator starts up, you will see a progress dialog appear in Android Studio.

8. Next, the emulator will appear and start its boot sequence.



```
ka   xample.tesla.myfirstapp;

or   d.support.v7.app.AppCompatActivity;
or   d.os.Bundle;

li   MainActivity extends AppCompatActivi

 @
 p   void onCreate(Bundle savedInstanceS
     onCreate(savedInstanceState);
     ntentView(R.layout.activity_main);
 }
```

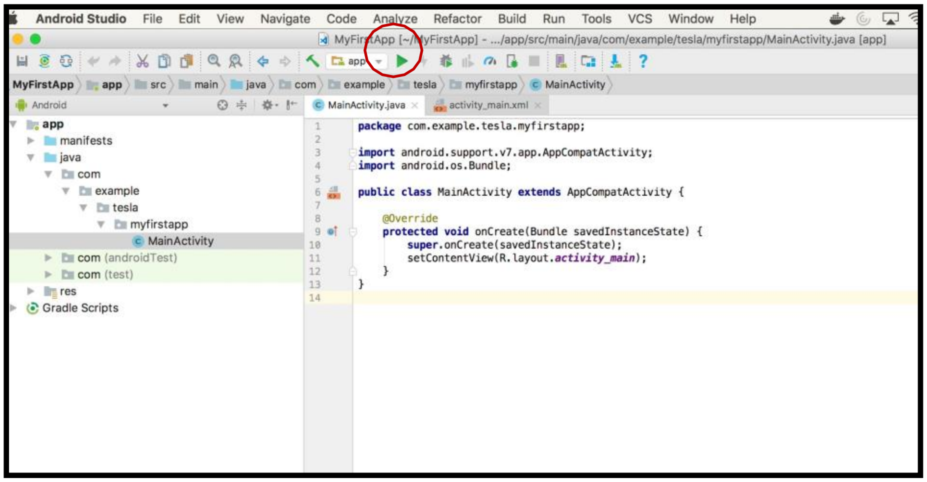
After the device has booted, the emulator will be ready for user interaction.

Part 4 – Running Your First App

In this part you will learn how to run the application you created in Part 2 in the Android Emulator you just created in Part 3.

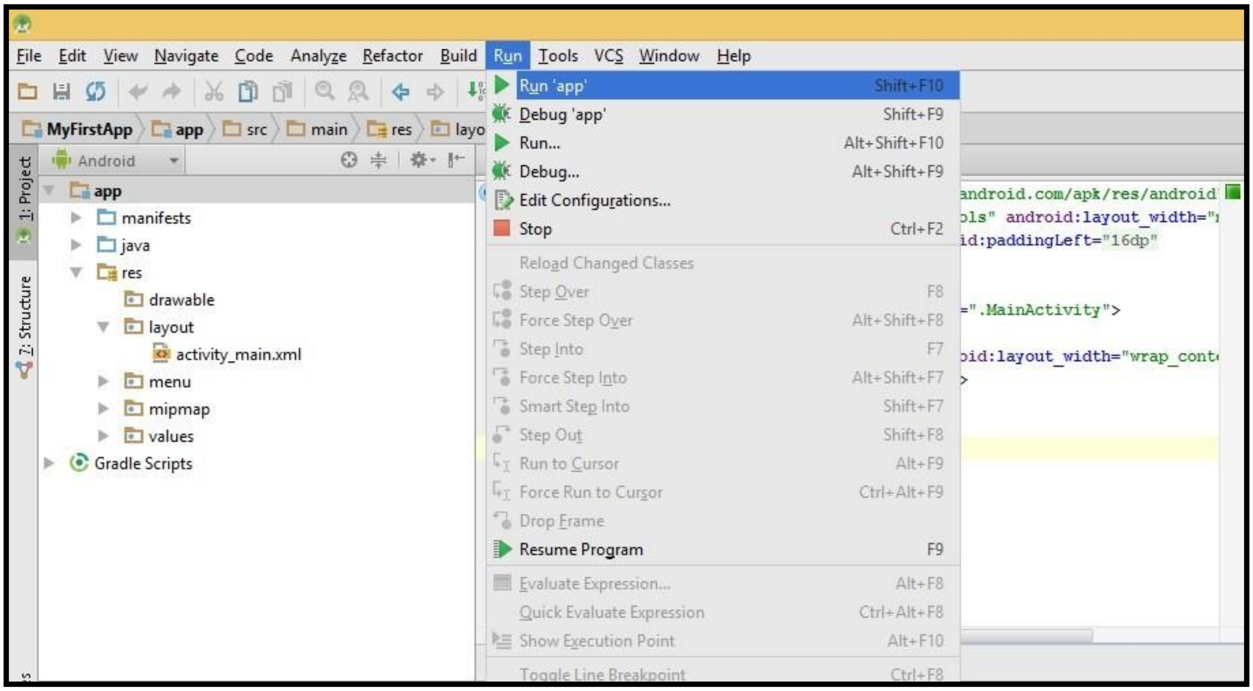
There are two ways to run the app:

Method 1: Return to Android Studio and simply click on the “Run ‘app’” Button

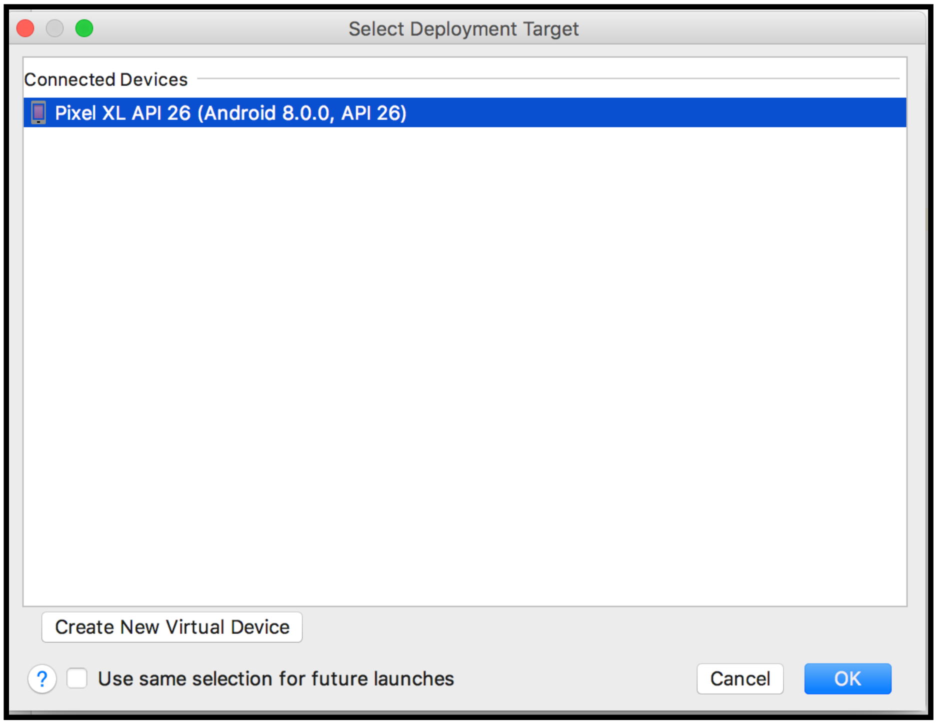


```
package com.example.tesla.myfirstapp;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Method 2: Return to Android Studio and select Run > Run ‘app’.

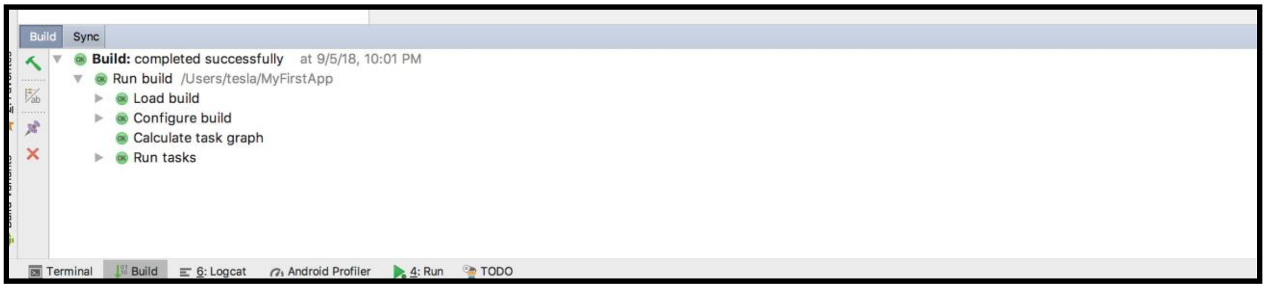


Next a window will pop-up to ask you to select which of your pre-configured AVD devices you would like to run the app on. If you do not have the correct SDK installed on your AVD for your app you will be prompted to install it. Select the Pixel XL you just created and click ‘OK’.


Select Deployment Target
Connected Devices
Create New Virtual Device
? □ Use same selection for future launches
Cancel
OK

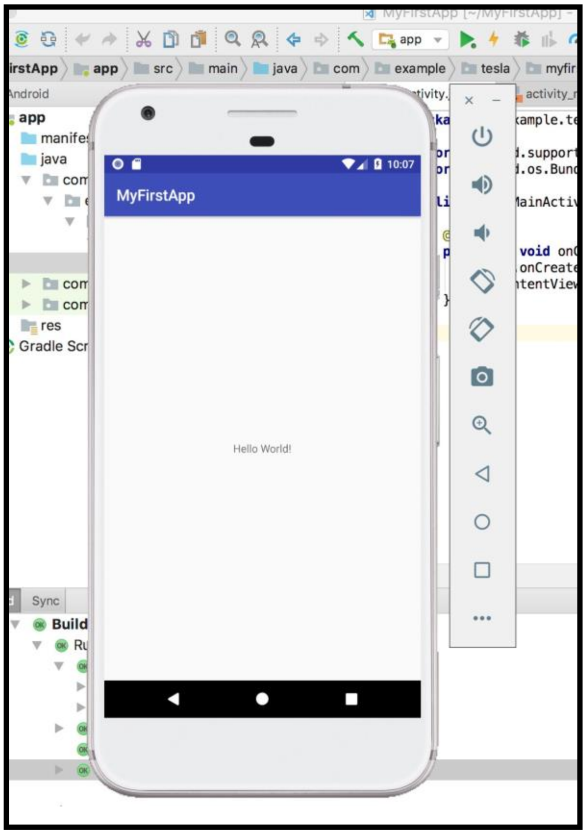
Pixel XL API 26 (Android 8.0.0, API 26)

In the Build Console panel, below the editor window, you will see output indicating that the application is being built, loaded and configured.



```
▼ Build: completed successfully   at 9/5/18, 10:01 PM
   ▼ Run build  /Users/tesla/MyFirstApp
      ▶ Load build
      ▶ Configure build
        Calculate task graph
      ▶ Run tasks
```

Return to your Emulator instance. You should now see your application, running in the Android Emulator, with the familiar ‘Hello World’ text.

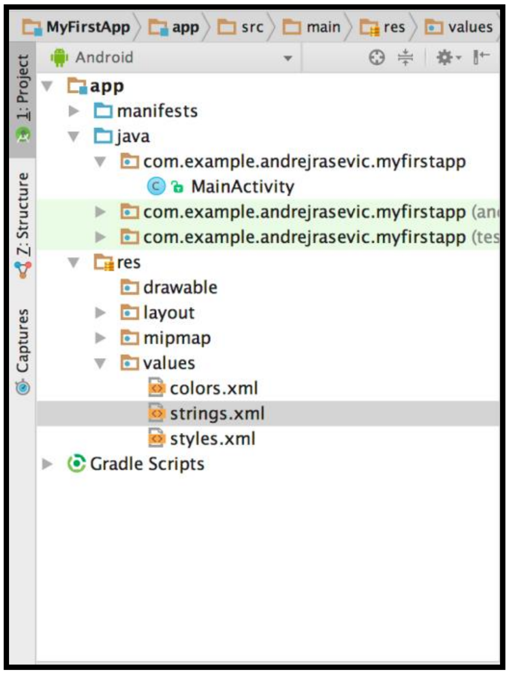

MyFirstApp [~/MyFirstApp]
app
irstApp
app
src
main
java
com
example
tesla
myfir
Android
tivity.
activity_
app
manife
java
com
com
com
res
Gradle Scr
ka
xample.te
or
or
.support
.os.Bund
li
MainActiv
@
p
void on
onCreate
ntentView
}
10:07
MyFirstApp
Hello World!
Sync
Build
Ru

Extra Challenge

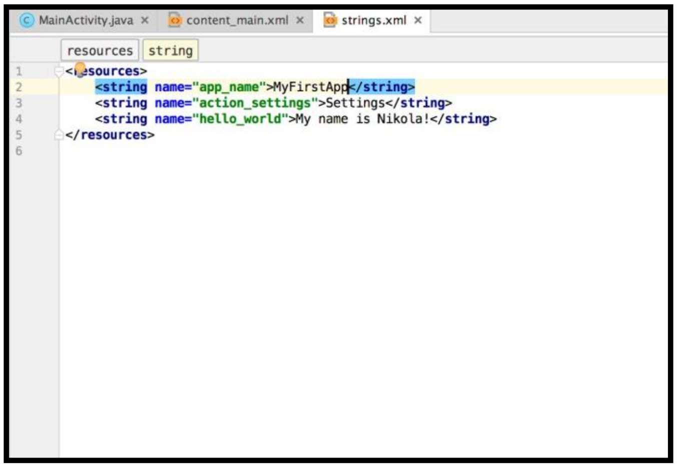
If you finish all the work above in class, then do the follow challenge activity as well.

1. **Modified Hello World** - Remember the first app you made? Let’s return to that!

2. In this part you’ll modify the original "Hello world!" message of your first app. To do this you need to modify the string value in \res\values\string.xml.

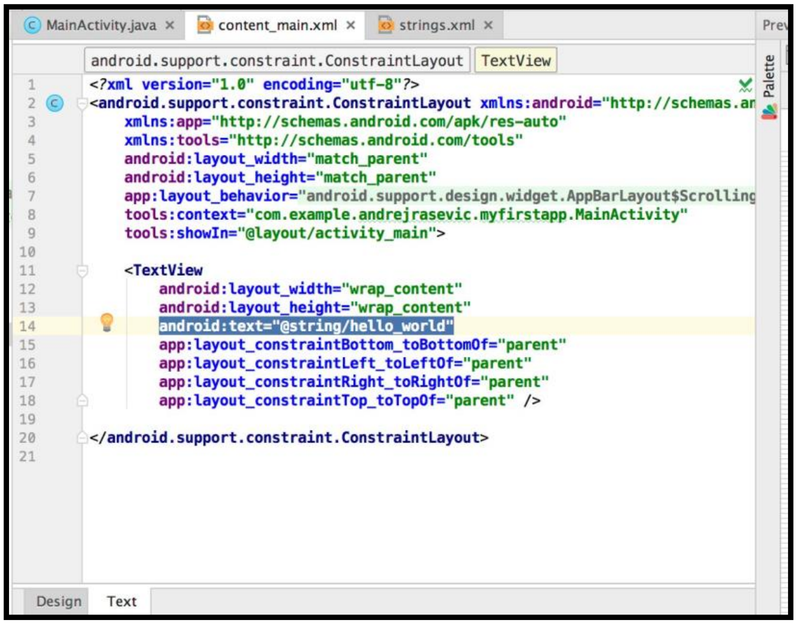


3. Add another string element with the text: “My name is <your_name>!”.



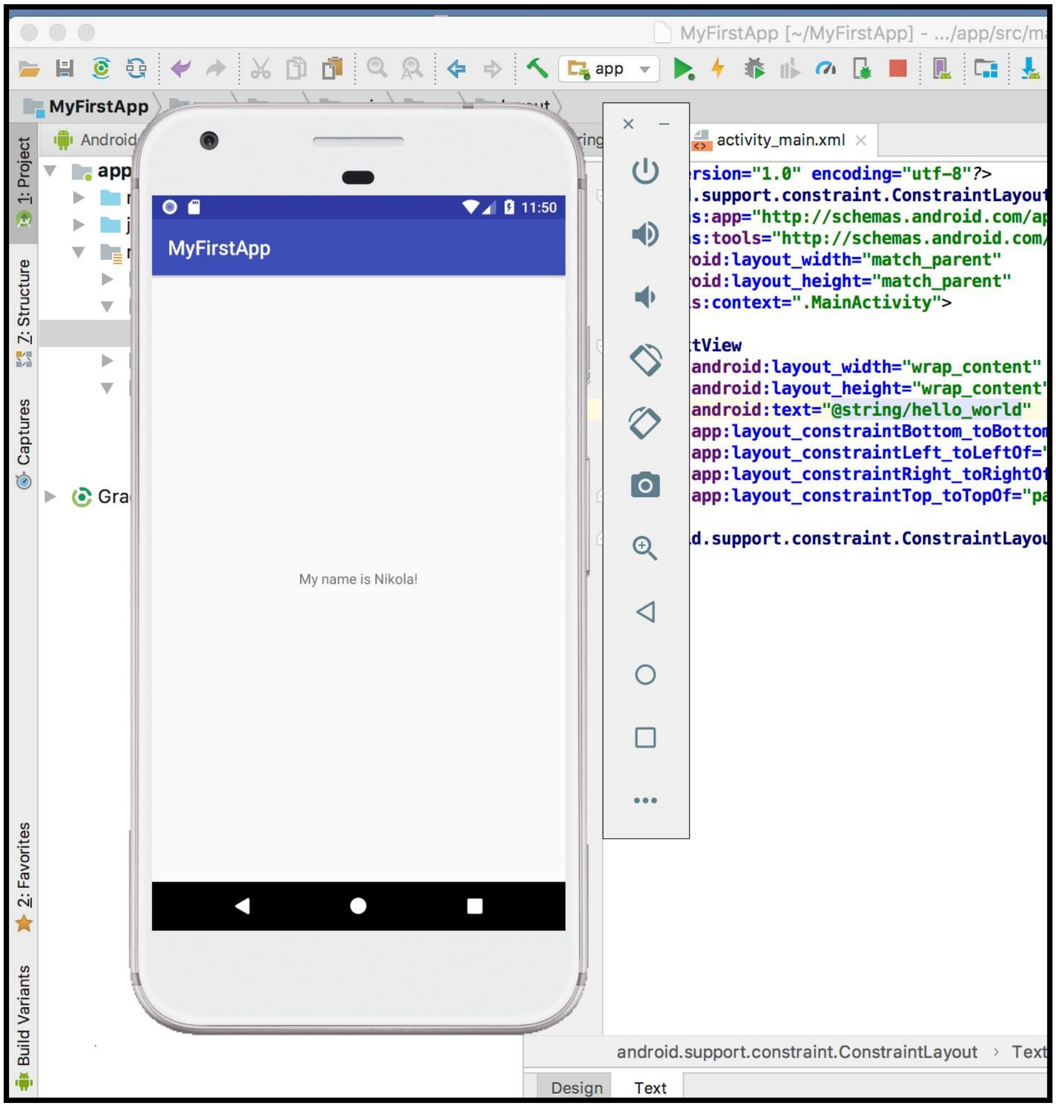
```
<resources>
    <string name="app_name">MyFirstApp</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">My name is Nikola!</string>
</resources>
```

4. Now go to the activity_main.xml file inside of res/layout. Edit the TextView element so it references the hello_world string element you just created in the previous step.



```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.an
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_behavior="android.support.design.widget.AppBarLayout$Scrolling
    tools:context="com.example.andrejrasevic.myfirstapp.MainActivity"
    tools:showIn="@layout/activity_main">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

5. Now run the app and see the change!



```
ersion="1.0" encoding="utf-8"?>
.support.constraint.ConstraintLayout
s:app="http://schemas.android.com/ap
s:tools="http://schemas.android.com/
oid:layout_width="match_parent"
oid:layout_height="match_parent"
s:context=".MainActivity">

tView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/hello_world"
app:layout_constraintBottom_toBottom
app:layout_constraintLeft_toLeftOf="
app:layout_constraintRight_toRightOf
app:layout_constraintTop_toTopOf="pa

d.support.constraint.ConstraintLayou
```

My name is Nikola!